Universiteit Leiden

Heuristic AI for fighting Hollow Knight-inspired boss fights

Tristan van der Poll (s3187977)

Supervisors:
Matthias Müller-Brockhausen & Giulio Barbero

Bachelor Thesis– Informatica

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 1, 2026

Abstract

Artificial Intelligence agents for digital action games must make decisions in dynamic adversarial environments with large and complex decision spaces. Heuristic agents, which use predefined rules to evaluate game states and select actions, offer a computationally lightweight and interpretable approach to this problem, avoiding the search cost that exhaustive methods struggle with as the branching factor of a game grows. Game AI research continues to expand into many specialised niches, and boss encounters represent one such niche: structured, scripted, yet genuinely challenging, and so far little explored as a target for heuristic agents. While heuristic methods have been applied to platform game navigation and combat strategy, no prior work, to the best of the author's knowledge, has evaluated heuristic agents on scripted boss encounters in a 2D action game.

This thesis investigates the design and evaluation of heuristic AI agents for boss encounters inspired by the 2D action platformer Hollow Knight. A custom simulation environment was implemented in Pygame, featuring two boss encounters of differing complexity. A series of six agents of increasing sophistication were designed and evaluated, ranging from a purely aggressive baseline to agents with full knowledge of boss states and attack patterns.

On the simpler encounter, Gruz Mother, the best performing agent achieved a win rate of 82.1%, demonstrating that balanced positioning and attack strategies are highly effective against a straightforward boss. On the more complex encounter, False Knight, agents without domain-specific knowledge failed to achieve a single win across 100.000 trials each, while an agent with domain-specific knowledge of boss states achieved a win rate of 86.1%. These results highlight the importance of domain-specific knowledge for complex encounters, and suggest that balanced general strategies form a strong foundation that can be extended with targeted domain knowledge.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Research Questions . . . 1
- 1.3 Thesis overview . . . 2

2 Related Work **2**

3 Background **3**
- 3.1 Heuristic AI for games . . . 3
- 3.2 Hollow Knight . . . 4

4 Methodology **4**
- 4.1 Research design and approach . . . 5
- 4.2 Environment implementation . . . 5
- 4.3 Boss fight design . . . 6
  - 4.3.1 Gruz Mother . . . 6
  - 4.3.2 False Knight . . . 7
- 4.4 Agent framework . . . 9
  - 4.4.1 Common agent interface . . . 10
  - 4.4.2 Input for agents . . . 10
  - 4.4.3 Action space . . . 11
  - 4.4.4 RandomAgent . . . 12
  - 4.4.5 SimpleAgent . . . 12
  - 4.4.6 AvoidantAgent . . . 12
  - 4.4.7 HybridAgent . . . 13
  - 4.4.8 PredictiveAgent . . . 14
  - 4.4.9 ReactiveKnowledgeAgent . . . 15
  - 4.4.10 AdaptiveKnowledgeAgent . . . 16
- 4.5 Experimental protocol . . . 17

5 Experiments **17**
- 5.1 Experiment design . . . 17
- 5.2 Evaluation Metrics . . . 17
- 5.3 Experimental setup . . . 19

6 Results **19**
- 6.1 Gruz Mother . . . 19
- 6.2 False Knight . . . 23

7 Discussion **26**
- 7.1 Gruz Mother . . . 26
- 7.2 False Knight . . . 27
- 7.3 Further Research . . . 28

8 Conclusions 29

References 31

1 Introduction

1.1 Motivation

Video games are a controlled and reproducible environment with clearly defined rules and measurable outcomes, making them well-suited for evaluating AI design. Games allow for a large number of trials to be conducted without physical cost or risk, unlike domains that require interaction with the real world. The structured nature of games also means that results are consistent and comparable across trials, which is important for drawing meaningful conclusions about agent performance. This reproducibility is not guaranteed across all areas of AI research; reported improvements in machine learning are sometimes found to not hold up once hyperparameters or random initialisations are varied, making it difficult to assert whether a claimed improvement is genuinely responsible for the result [8]. Furthermore, many games are designed with increasing complexity as they progress, providing a natural framework for incremental research, where agents can be evaluated against increasingly difficult challenges.

Action games in particular introduce challenges that closely mirror real-world AI problems. These games require agents to make decisions within large, dynamic adversarial environments, and often operate with incomplete information about the game state. These properties make action games a compelling testbed for AI systems that need to act quickly and effectively under uncertainty and complexity [19].

Within action games, boss encounters represent a particularly interesting case for AI research. These encounters are structured and rule-based, consisting of defined attack patterns and phases. However, these boss encounters remain dynamically challenging, requiring an agent to balance offensive and defensive decisions across a large space of possible boss states and player responses. This structure makes boss encounters a controlled environment for evaluating AI decision-making, while still presenting a genuine challenge.

Given the large decision space and structured but reactive dynamics of 2D boss encounters, heuristic-based decision systems provide a practical, computationally efficient and interpretable starting point for investigation. This thesis therefore focuses on the design and evaluation of heuristic AI agents for Hollow Knight-inspired boss fights [18].

1.2 Research Questions

Effective AI design for games with large, complex decision spaces requires both a clear methodology for evaluation and an understanding of the design decisions that lead to better performance. This thesis addresses the following main research question:
How can the design of decision-making strategies for game AI be optimised for dynamic, scripted boss encounters?

To answer this question, two subquestions are investigated. The first subquestion addresses comparing agents requiring a meaningful evaluation framework beyond a single metric:

How can the performance of different decision strategies be measured and compared in the game Hollow Knight?

Second, designing effective agents requires balancing competing objectives, dealing damage efficiently while avoiding incoming attacks. Understanding these trade-offs is essential for improving agent design:
What trade-offs exist between competing performance objectives when designing decision strategies for dynamic combat encounters?

Together, these subquestions build towards the main question by establishing how performance is measured and what design principles lead to better agents.

1.3 Thesis overview

This thesis is structured as follows. Chapter 2 discusses related work. Chapter 3 provides the necessary background on heuristic AI and boss encounter design. Chapter 4 describes the simulation environment, boss designs, and agent implementations. Chapter 5 outlines the experimental setup and evaluation metrics. Chapter 6 presents the results and Chapter 7 discusses these results, draws conclusions, and suggests future research.

2 Related Work

Boss encounters are a common design pattern in action games, typically featuring a powerful enemy with patterned attack sequences, phase-based behaviour changes, and clearly defined player abilities, that the player must learn and respond to in order to progress. These boss encounters represent structured yet challenging sections of digital games. Siu, Butler, and Zook propose a formal programming model for boss encounters in 2D action games, describing them as structured state-driven systems with predictable yet reactive dynamics [14]. This formalisation suggests that boss fights can be conceptualised as controlled adversarial systems, in which an agent must balance offensive and defensive actions.

In dynamic combat settings, heuristic-based AI systems are an attractive design approach. Heuristics allow agents to evaluate game states, using predefined rules or evaluation functions, such as prioritising damage avoidance, maintaining safe positions, or exploiting attack windows. Similar approaches have been applied in platform game environments, where agents reuse predefined movement skills to navigate complex levels with limited computation [4]. Compared to simulation-based methods, heuristic systems are computationally lightweight, and well-suited for combat settings with large decision spaces like boss encounters.

Heuristic approaches have been applied to games beyond the platformer genre. Churchill and Buro describe a heuristic search architecture for combat scenarios in real-time strategy (RTS) games, where agents must make decisions in large action spaces [3]. Their work demonstrates that heuristic search can outperform hand-crafted combat scripts, highlighting the effectiveness of structured rule-based decision making in adversarial game environments. However, RTS combat differs significantly

from 2D action boss encounters: the game state, action space, and required decision granularity are fundamentally different. Boss encounters present a more constrained but more reactive combat scenario, where timing and positioning relative to a single powerful enemy are the primary challenges.

Evaluating game AI performance requires metrics that capture more than a single dimension of performance. The General Video Game AI (GVGAI) framework, described by Perez-Liebana et al., provides a multi-track evaluation methodology for comparing agents across a wide range of games, using metrics such as win rate, and score as primary indicators [7]. Bravi et al. extend this further, proposing a wider set of metrics for evaluating decision-making in game AI agents, arguing that a single metric is insufficient to capture the full picture of agent performance [2]. This research follows a similar philosophy, evaluating agents across multiple metrics including win rate, survival time, damage dealt, damage taken, and closeness to victory, to provide a more complete picture of agent performance than win rate alone would allow.

While heuristic approaches have been applied to combat scenarios and platform game navigation, and evaluation frameworks for game AI are well established, heuristic combat AI for 2D platformer enemies specifically has received only limited attention. Promsutipong and Kotrajaras evaluate a finite-state-machine and rule-based heuristic agent's battle performance against enemies in a 2D action platform game, finding that it performs similarly to human players [9]. This demonstrates that heuristic combat agents are a viable approach in this genre, but heuristic agent evaluation specifically has not yet been extended to structured, multi-phase encounters. Hollow Knight itself has previously been used as a deep reinforcement learning benchmark, with agents evaluated specifically on its boss fights [20]. However, to the best of the author's knowledge, no prior work has evaluated heuristic, rule-based agents on these similarly structured boss encounters. Existing work in adversarial combat AI, such as Tang et al.'s evolutionary approach to fighting games [17], demonstrates that competitive performance is achievable against a skilled, reactive opponent, but focusses on symmetric two-player combat rather than the asymmetric player-versus-boss dynamic studied here. This research addresses that gap by designing and evaluating a series of heuristic agents of increasing complexity against two Hollow Knight-inspired boss encounters, contributing both a novel simulation environment and an empirical comparison of heuristic decision strategies in scripted combat.

3 Background

3.1 Heuristic AI for games

Artificial Intelligence in digital games must operate in dynamic and adversarial environments where the space of possible game states and player-boss interactions is large. In contrast to domains with small, well-defined state spaces, action games involve deep and wide decision trees: many possible positions, attack timings, and state combinations. Exhaustive search methods like MCTS scale poorly as this branching factor grows, requiring significant computation to evaluate even a fraction of possible futures. Heuristic methods avoid this cost entirely by relying on predefined rules rather than search, making them both easier to design for a specific domain, and more computationally

efficient in large or deep decision spaces.

Heuristics are rules or evaluation functions that guide decision making systems without making use of an exhaustive search [12]. An example could be "Move away if the enemy is within X distance". Combining these rules creates a lightweight agent that can determine an action by checking which heuristics apply to the current game state and acting accordingly.

This makes heuristic agents computationally cheap, interpretable and consistent in their behaviour. It is easy to reason why an agent made a particular decision, which is valuable for research and evaluation. This stands in contrast to deep learning and reinforcement learning approaches, where the complexity of the underlying model obscures how a decision was reached. Dedicated explainability models have been developed, to address this, but these typically only offer partial insight into an agent's behaviour [10].

However, heuristics have limitations. Designing good heuristics for an AI agent requires domain expertise, and heuristics do not learn or adapt to situations outside of their predefined rules.

Boss encounters, structured fights against powerful enemies with defined attack patterns, are a particularly natural fit for heuristic agents. Their predictable and rule-based nature aligns well with heuristic design, and the low computational cost of heuristics makes them well-suited to the large, branching decision spaces these encounters present.

3.2 Hollow Knight

Hollow Knight [18] is a 2D action platformer developed by Team Cherry, known for its precise combat and its numerous and well-designed boss encounters. The player has access to a limited, but flexible moveset, consisting of simple movement, jumping and directional attacks. Enemies are fought in a real-time environment, and boss encounters are a central part of the game, each with distinct attack patterns and phases.

The boss encounters in Hollow Knight are well structured, with clearly defined movesets, see Figure 2 and Section 4.3.1 for an example, making them a natural fit for the kind of controlled evaluation this research requires. The game also has a range of boss difficulty, from simple early encounters to complex multi-phase fights, which supports incremental research.

4 Methodology

This section covers the design and implementation of the simulation environment, the boss encounters, and the agents used in this research. First, the research design and overall approach are outlined, followed by a detailed description of the environment and boss implementations. Finally, the agent framework and experimental protocol are described.

4.1 Research design and approach

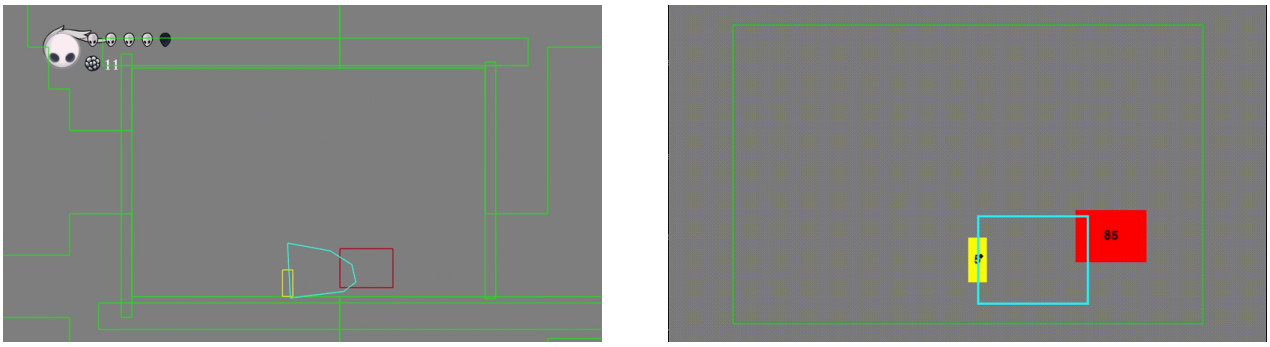
This research investigates the performance of heuristic agents in structured boss encounters inspired by Hollow Knight as mentioned in Section 3.2. To evaluate these agents in a controlled and reproducible setting, a custom simulation environment was built using Pygame [11], rather than modifying the original game. This approach offers several advantages: the environment can be simplified to the essential mechanics, and the simulation can be accelerated to run hundreds of thousands of trials in a short amount of time, enabling robust data collection.

Development followed a deliberate progression. The simulation environment was built first, establishing the foundation for all subsequent work, as described in Section 4.2. Gruz Mother was then implemented as an initial boss encounter, described in Section 4.3.1. Her simple attack patterns and lack of phases make her a suitable starting point for agent development and evaluation. After establishing a set of agents for Gruz Mother, False Knight was implemented as a second, more complex boss encounter, described in Section 4.3.2. False Knight offers additional obstacles, more numerous and complex attacks, and phases, providing a more demanding test bed and allowing for an evaluation of whether agents can generalise across encounters or require domain-specific knowledge to perform well. The agents were designed in order of increasing complexity, each building on the limitations of the previous, as described in Section 4.4. This progression, from a purely random baseline to agents with full knowledge of the boss state, allows for a structured comparison of different heuristic approaches.

4.2 Environment implementation

The environment was created using Pygame, to replicate similar physics to Hollow Knight's physics. The recreation of these physics and boss behaviours is based on subjective feel rather than access to the game's source code or exact internal data. Boss behaviour and movement were informed by the Hollow Knight Wiki entries on Gruz Mother [6] and False Knight [5], as well as a hitbox visualisation video by Skurry [15]. Additionally, the implementation reflects the author's extensive experience with the game, accumulated over 892 hours of gameplay with a strong focus on boss movement patterns, including hitless completion of Pantheons 1 through 4 (gauntlet-style boss sequences completed without taking damage).

When running this simulation, an agent or a human player needs to be selected to control the simulation. When a human player is selected, the human player uses keyboard inputs to control the in-game character, when an agent is selected, the agent will be asked for an action. The arenas for the boss fights are squares, similar to the arenas from the game. As shown in Figure 1b, the environment is kept simple, only showing the hitboxes of all actors in the scene. This is to ensure the simulation remains clear and similar to the hitboxes of the original game. Player and boss movement have been tweaked to match the physics of the original game, and provide the same feel to a human player. Upon taking damage, the player is granted invincibility time, and is knocked back from their position, which matches the behaviour in Hollow Knight. Similarly, landing an attack on a boss causes a slight recoil effect on the player, pushing them back from the point of contact. This recoil is exaggerated on downward attacks, allowing the player to pogo on enemies.



(a) Gruz Mother fight in Hollow Knight, stripped to hitboxes.

(b) The Gruz Mother simulation built for this research.

Figure 1: Comparison of Hollow Knight's hitbox visualisation and the simulation environment.

The player has five masks, which represent the player's health. Most attacks, or contact damage sources, deal one mask of damage. Each attack from the player deals 5 units of damage to enemies, including bosses, who have a set number of health points. This number of health points is usually set to a number a lot higher than the player's attack damage, requiring a large amount of hits to defeat. An enemy whose health reaches 0 is defeated, similar to how the player's masks being depleted to 0 ends the simulation. Charms, which in Hollow Knight provide a wide range of buffs and modifications to the player abilities, are excluded from this simulation. Their inclusion would significantly expand the action space and introduce a large number of variables, making controlled evaluation of heuristic agents considerably more difficult.

Every frame, the simulation updates in a fixed order to ensure consistency between player, boss and collision detection. The game state is generated first, giving the agents and boss an accurate snapshot of the environment before any decisions are made. The boss then acts on this state, followed by the player, ensuring that both react to the same frame of information. Damage and collision are resolved after both have moved, completing the update cycle.

4.3 Boss fight design

4.3.1 Gruz Mother

Gruz Mother is a simple boss encounter. She serves as an early skill check for the player. As a result, Gruz Mother is not a very complicated boss encounter, and does not demand a lot of difficult movement or strategy from a player. This simplicity makes the Gruz Mother an ideal baseline encounter: straightforward enough for initial heuristic agent development, yet serving as a meaningful reference point when the same agents are later shown to be insufficient against the more complex False Knight encounter, as seen in Section 6.2.
The boss has two moves in its moveset, and an idle state. Her only way of dealing damage is contact damage, meaning overlapping with the player results in the player taking damage. In Figure 2, a state machine of Gruz Mother's behaviour can be seen. Gruz Mother is idle during an attack cooldown of three seconds. During this idle state, Gruz Mother flies aimlessly through the

arena. Once the three-second cooldown is over, Gruz Mother makes a random choice between three options. Either she continues flying for another three seconds, restarting the loop, or she chooses one of her attacks.

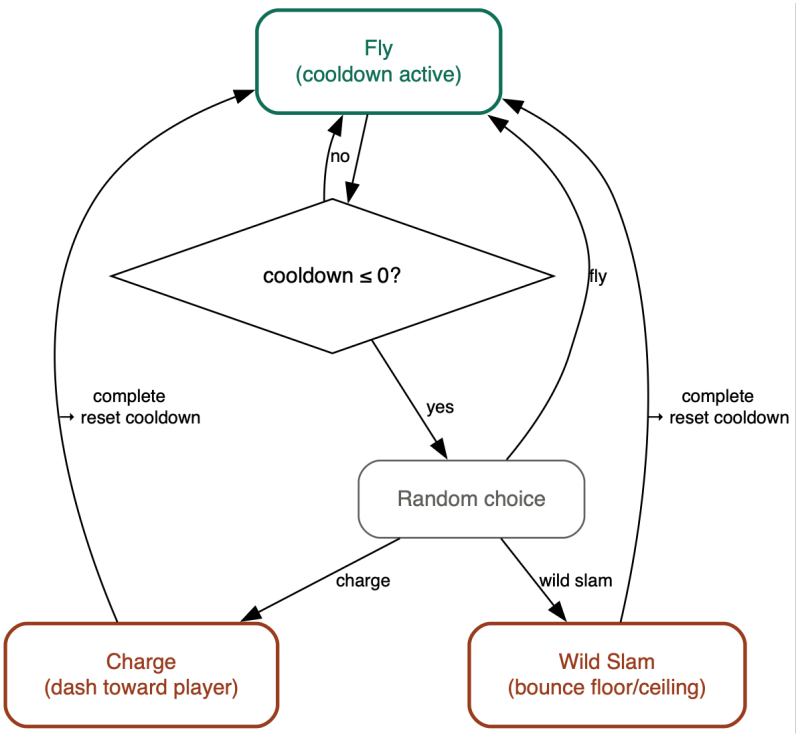


Figure 2: Behaviour state machine for Gruz Mother

The first attack is the charge attack, where Gruz Mother backs up a bit, and then flies in a straight line towards the player for a set distance. If Gruz Mother hits a wall during this attack, she bounces off the wall slightly, and ends the attack there, and goes back to flying. If she reaches the set distance, she stops the attack and goes back to the idle state. The player will need to dodge this straight flight path or get hit and take one mask of damage.

Gruz Mother's other attack is the wild slam attack. She quickly starts bouncing up and down against the floor and ceiling. She bounces between 15 and 17 times, and moves with a diagonal path up and down, moving forward slowly. When hitting a horizontal wall, she turns around and continues the attack. The player is expected to pass underneath Gruz Mother as she moves up, or jump over her when she moves down. As with all attacks, she deals one mask of damage as contact damage.

To win the fight, the player simply needs to deplete Gruz Mother's health of 90 points. This is the equivalent of hitting Gruz Mother with an attack 18 times.

4.3.2 False Knight

False Knight is the first real challenge the player faces in Hollow Knight. This boss is required for progression, blocking off a path to a useful ability. As such, False Knight is a more complicated

boss encounter, making use of more attacks, and different phases of the fight. This design forces the player to carefully consider strategies that balance dealing damage and playing more defensively, a skill that is essential later in the game. False Knight is a maggot in a large suit of armour, much larger than the player's size, giving him a large presence. This large presence comes with a large hitbox, and is therefore easy to deal damage to, making the fight easier for players who have just started the game.

The fight starts in an idle state for a random amount of time, between one and two seconds. During this idle state, False Knight behaves very differently from Gruz Mother. False Knight stands still, waiting before making a decision. He chooses an attack from the list of attacks available, and performs that attack. After that, False Knight goes back to the idle state for another random amount of time between one and two seconds.

The available attacks depend on several specific variables. The state machine determining False Knight's behaviour can be seen in Figure 3. The attacks are the following:

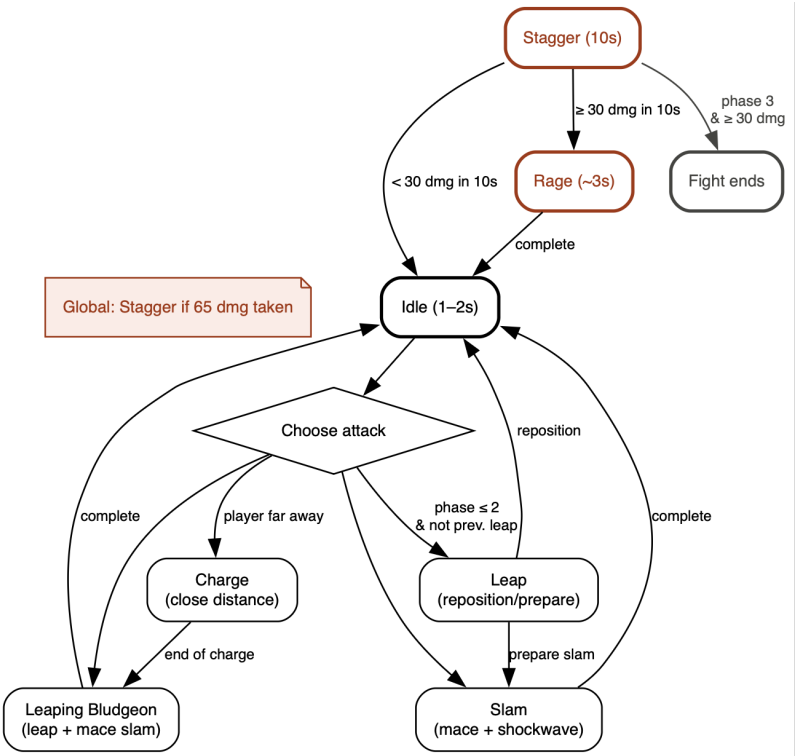


Figure 3: Behaviour state machine for False Knight

Leap: A large jump from False Knight's current position. The destination depends on the type of Leap attack performed. This attack can either be a random leap for repositioning, or can be a leap to prepare for a Slam attack right after, which is then a forced attack. In this last case, False Knight jumps away from the player to the corner, to maximise the shockwave of the Slam attack. False Knight will stop using Leap in phase three.

Slam: False Knight slams his mace down onto the ground, dealing two masks of damage. After this slam, a shockwave is sent towards the other side of the arena, growing in speed and horizontal size as it travels further, stopping when hitting the wall of the arena. This shockwave deals only 1 mask of damage. During phases two and three, the slam spawns falling rocks from the ceiling, as an extra obstacle for the player to dodge.

Leaping Bludgeon: False Knight leaps into the air, and aims to land in front of the player's position from when the leap began. Upon landing, False Knight slams the mace down in front of him. This mace attack deals 1 mask of damage. In phase three, this attack also spawns falling rocks from the ceiling.

Charge: When the player is too far away from False Knight, he may choose to run at the player to close the distance, and once he gets close enough, he performs Leaping Bludgeon.

Rage: When 65 points of damage are dealt to the armour of False Knight, he is staggered, and the maggot inside of the armour emerges from the armour. During this, the player can deal damage to the maggot. If enough damage is dealt within 10 seconds of the maggot emerging, the maggot enters the armour again, and False Knight enters Rage. This also starts the next phase of the fight. False Knight leaps to the centre of the arena. He starts violently slamming his mace into the floor alternating between left and right. These are not the same as the Slam attack, as no shockwave is caused. This attack does always spawn more falling rocks. After about three seconds of this alternating slamming, False Knight goes back to the idle state, and continues the fight. If not enough damage was dealt within the 10 second time, False Knight stands back up, and continues the fight as normal, without the phase changing, and the fight therefore does not progress.

As mentioned in the description of the Rage attack, False Knight gets staggered once 65 points of damage, or 13 slashes, are dealt to the armour. The maggot has 30 health points; if this damage is dealt, the phase increases and the Rage attack begins. In phases two and three, falling rocks emerge from the ceiling on certain attacks. These falling rocks can be hit by the player, which redirects them in a straight line in the direction they were hit into. If the rock collides with False Knight, it deals 10 damage, the equivalent of two slashes from the player.

The fight is completed once False Knight has been staggered three times, and the maggot has received his last 30 damage, after which the fight is complete. In the original game, after the third stagger, False Knight enters Rage one more time, after which a Leaping Bludgeon towards a specific arena target is performed, breaking the floor, and permanently staggering False Knight, letting the player fall down and deal the remaining damage to the maggot and complete the fight. In a later version of the fight, introduced in the Godmaster DLC update, False Knight does not break the floor and the fight simply ends after the third stagger. This paper uses the Godmaster iteration the fight.

4.4 Agent framework

Seven agents were designed and implemented for this research, presented here in order of increasing complexity. Each agent builds on the limitations of the previous, with the goal of progressively

improving performance. The first 5 agents (RandomAgent, SimpleAgent, AvoidantAgent, HybridAgent and PredictiveAgent) were designed for Gruz Mother. The final two agents were designed to make use of additional game state information specific to False Knight. Each agent is described in terms of its design philosophy, core strategy, positioning rules, attack rules and limitations. This structure reflects the layered nature of heuristics design. The philosophy motivates the strategy, the rules implement it, and the limitations justify the next agent in the sequence. The portrayed limitations for each agent are based on experiments run described in Section 5.

4.4.1 Common agent interface

All agents are subclasses of the Agent class. The Agent class has a function that is overridden by the subclasses, determining what action the agent will perform. Each agent returns an action directly to the simulation, which applies it to the player. The action space available to agents is described in Section 4.4.3.

4.4.2 Input for agents

Agents need to see what happens in the environment, or else they have nothing to base a decision on. During each frame of the simulation, the game state is generated, which is a dictionary of information that agents use to make decisions. The metrics contained in the game state can be found in Table 1.

Metric Name	**Datatype**	**Range**
Player health	int	0 to 5
Boss health	int	0 to 90 or -5 to 285
Player global position	Vector2 float	Range defined by map boundaries
Boss global position	Vector2 float	Range defined by map boundaries
Player size	Vector4 float	25 units wide, 60 units tall
Boss size	Vector4 float	95 units wide, 70 units tall, or 193 units wide and 222 units tall
Boss velocity	Vector2 float	$(-\infty, -\infty)$ to (∞, ∞)
Player attack cooldown	float	0.0 to 0.5333
Arena walls	int	Defined by map boundaries

Table 1: Game Metrics Specifications

For the agents designed for the False Knight fight, the game state includes extra metrics, so agents can keep track of falling rocks, shockwaves, and incoming attacks. This means that for those agents, there is more information added to the game state, which can be found in Table 2.

Metric Name	Datatype	Range
Boss State	Enumerator	Fly, Wild Slam, and Charge for Gruz Mother. Idle, Slam, Rage, Stagger, Leap, Charge, and Leaping Bludgeon for False Knight
Phase	int	1 to 3
Maggot position	Vector2 float	Range defined by map boundaries
Shockwave position	Vector2 float	Range defined by map boundaries
Shockwave direction	int	-1 or 1
Shockwave edges	float	Range defined by map boundaries
Falling rock positions	Vector2 float	Range defined by map boundaries
Mace	Vector4 float	338 units wide, 148 units tall (Slam, Rage) or 189 units wide, 430 units tall (Leaping Bludgeon)

Table 2: Additional information added to the game state during False Knight simulations

Agents can use all of this information to inform their decisions, for example, determining the distance between the player and the boss, detecting when particular situations arise, and selecting an appropriate response.

4.4.3 Action space

The action space consists of six booleans: directional movement (left, right, up, down), jumping, and attacking. These inputs can be combined simultaneously, subject to priority rules: when left and right are both active, left takes precedence, and when up and down are both active, up takes precedence. The up and down inputs are only meaningful when attack is also active, as they determine the vertical direction of a slash rather than player movement. Additionally, a downward slash is only valid when the player is airborne. These rules mean the effective number of distinct, meaningful action combinations is smaller than the $2^6 = 64$ raw boolean combinations would suggest. An example action is shown in Table 3, representing a leftward jump with a horizontal slash. When the attack component is set to True, the player attacks in the direction the player is facing, with right and left determining that direction, and up and down causing the player to slash vertically instead of horizontally. The left and right direction is remembered if no horizontal movement is entered, but for vertical slashing, the up or down component must be True, or the player defaults to horizontal slashing.

left	True
right	False
up	False
down	False
jump	True
attack	True

Table 3: An example of an action list, representing an action that can be performed. Here, the action performed is a jump to the left with a slash.

4.4.4 RandomAgent

The RandomAgent is the simplest agent. This agent was implemented with the goal of being a baseline for other agents to compare to. The RandomAgent has a dictionary of the action space. It also has the global seed, which is used upon initialisation to initialise the randomisation. Every 0.15 seconds, the RandomAgent randomly chooses an action from the action space to perform. In between choices, that choice remains until the next decision is made.

4.4.5 SimpleAgent

Design Philosophy The SimpleAgent is a highly aggressive agent, designed to maximise damage output with no regard for own safety.

Target boss The SimpleAgent was designed for the Gruz Mother fight.

Core strategy The SimpleAgent moves towards the boss and attacks when in range, attempting to remain in range to get as many attacks to hit the boss as possible.

Positioning rules In between attacks, the SimpleAgent chooses to always move horizontally towards the boss's centre.

Attack rules The SimpleAgent slashes upwards when the boss is overhead, and sideways when horizontally in range. Given the positioning rule of moving horizontally towards the boss, this means that the agent usually resorts to upwards slashing, as the agent will match the horizontal position of the boss.

Limitations The SimpleAgent is designed to deal as much damage as quickly as possible, with no regard for incoming damage. This disregard means it reliably takes heavy damage, making it a minimal baseline better than the RandomAgent, but a poor fighter.

4.4.6 AvoidantAgent

Design Philosophy The AvoidantAgent is an overly defensive agent, opting to move to a safe distance from the boss, and only land occasional slashing. The AvoidantAgent makes use of a

'panic jump', jumping when the boss gets too close and the agent gets cornered, in an attempt to gain more distance, and potentially jump over the boss.

Target boss The AvoidantAgent was built for the Gruz Mother fight.

Core strategy The AvoidantAgent prioritises survival over damage output. The agent attempts to maintain a safe distance from the boss at all times, retreating when the boss gets too close. When cornered, the agent uses a panic jump to escape, accepting the risk of contact damage in exchange for regaining distance. Damage is not actively sought, making the AvoidantAgent the opposite extreme of the SimpleAgent.

Positioning rules The AvoidantAgent attempts to stay in the middle of the arena, to avoid getting cornered. If the boss gets too close, the AvoidantAgent moves to a safe distance, to avoid taking contact damage. When the AvoidantAgent has been driven into a corner by constantly moving to a safe distance, the AvoidantAgent will eventually be too close, and a panic jump will be used to get away and hopefully jump over the boss.

Attack rules The AvoidantAgent will not seek to attack the boss. The only offensive output is an opportunistic pogo when airborne from a panic jump.

Limitations The AvoidantAgent's tendency to move to the centre of the arena can move the agent into danger, which can be in conflict with the strategy of staying at a safe distance. If the boss approaches too quickly, the agent might not have enough time or space to properly avoid the attack. The agent also does not seek to attack, which drags out the fight and likely will not be able to deal enough damage before taking fatal damage.

4.4.7 HybridAgent

Design Philosophy The HybridAgent is a mix of the SimpleAgent and AvoidantAgent, with some added improvements. The HybridAgent should attempt to move to a range close enough to deal damage, but not too close to take too much damage. Staying in this safety window allows the agent to deal a lot of damagewhile avoiding the heavy damage that the SimpleAgent takes.

Target boss The HybridAgent was built to fight Gruz Mother.

Core strategy The HybridAgent moves horizontally towards the boss, but stops when in range of a horizontal slash. The agent then attempts to maintain this spacing, by moving further away if the distance to the boss becomes smaller, and moving closer if the distance becomes larger. This should allow the agent to attack regularly similar to the SimpleAgent, but at a safe distance.

Positioning rules Similar to the AvoidantAgent, the HybridAgent will also attempt to escape when backed into a corner, with the same 'panic jump', and with a pogo to gain more height and therefore hopefully make it over the boss. The agent will also attempt to move towards an ideal

range of horizontal distance allowing for attacking the boss, but not taking damage too frequently like the SimpleAgent.

Attack rules The HybridAgent inherits most of its attack rules from the SimpleAgent, attacking when in horizontal distance, attacking upwards when the boss is above, and using a pogo attack when the boss is below, which is from the AvoidantAgent. Next to this, the HybridAgent also has a new attack, the jump attack. This is to also be able to deal damage when the boss is diagonally above the player, giving the HybridAgent more opportunities to deal damage quickly.

Limitations The HybridAgent does not take into account what direction the boss is going in, making it tough to determine when the right moment is to escape from out of the corner of the arena, or what the correct decision is to escape, to jump or to walk underneath. This means that the HybridAgent almost always takes damage every time it is driven into a corner. The HybridAgent is also very focussed on the jump attacks, which makes the agent not prioritise horizontal distance enough, and therefore can occasionally take contact damage.

4.4.8 PredictiveAgent

Design Philosophy The PredictiveAgent is similar to the HybridAgent. The PredictiveAgent makes use of the boss's previous position to attempt to determine the direction and speed, and predict where the boss is moving. This would allow the agent to determine when safe windows present themselves for escaping when cornered, and when the boss may be charging at the player. This prediction should allow the agent to avoid more effectively, and dodge attacks better.

Target boss The PredictiveAgent was designed for the Gruz Mother fight.

Core strategy Using the boss's previous position, and getting the difference with the current position, the agent can figure out where the boss has moved and how fast. Using this information, the agent can determine when the boss is performing the wild slam attack, which involves fast vertical movement. Using this, the agent can determine if the boss is moving up or down, and predict what the better escape strategy is, either walking over the floor if Gruz Mother is above and moving up, or jump and pogo and move over if Gruz Mother is moving downwards. This should allow for a more effective dodging strategy, and should allow the agent to survive for longer, and therefore succeed more often.

Positioning rules The positioning rules are similar to the HybridAgent, with the only exception being the predictions for the wild slam attack, and using that information to determine whether the agent should jump or not when escaping from the corner.

Attack rules The attack rules are identical to the HybridAgent, except for the use of the determined boss movement to tell when a jump attack is warranted, which makes the agent more mindful of the attack cooldown. This also allows the agent to move away more, creating more distance, lessening the contact damage due to not moving enough.

Limitations The PredictiveAgent does not deal with charge attacks well. These attacks are hard to determine with the previous position strategy, as the charge is high speed, but not a set orthogonal direction. This makes it difficult to determine what should be done. This causes the most damage, as the agent does not know how to deal with a charge very well. The PredictiveAgent also does not perform well at all against other bosses than Gruz Mother, only knowing her patterns.

4.4.9 ReactiveKnowledgeAgent

Design Philosophy The ReactiveKnowledgeAgent is an agent built to make use of perfect knowledge. This means that added information in the game state is used, allowing the agent to fight False Knight more effectively than any other agent so far. The agent can use the state property of the False Knight to determine when the boss is idle, which is a good opportunity to deal a lot of damage, or when the boss is attacking, and what attack that is. False Knight does not have many clear signs showing what attack will occur, so using the state will allow the agent to make decisions accordingly.

Target boss The ReactiveKnowledgeAgent was built for False Knight.

Core strategy The ReactiveKnowledgeAgent has different rules for each of False Knight's attacks, allowing the agent to react to the changes in state. Using the state allows the agent to decide when to deal damage, or when and how to dodge.

Positioning rules The ReactiveKnowledgeAgent attempts to position itself close enough such that the edge of its attacks will overlap with the edge of the boss. During each of False Knight's attacks, the agent will position differently to avoid damage or to continue dealing damage. During the Leap attack, the agent waits and loosely follows when False Knight is far enough. For the Slam attack, the agent times a jump to perfectly jump over False Knight's mace if it is close enough. Otherwise, the agent notices the shockwave, and can jump over that instead, and moves towards False Knight to be able to deal more damage. For the Leaping Bludgeon attack, the agent moves underneath False Knight, and then moves out to the opposite side to avoid the downwards slam. During Rage, the agent walks to the corner during the jump into the middle, and moves back towards the centre in an attempt to be able to hit a falling rock towards False Knight.

Attack rules The ReactiveKnowledgeAgent attempts to stay within range for as long as possible, ensuring that it can always attack when the attack cooldown is finished. During each attack, the agent attempts to deal damage, and during an idle state, the agent tries to maximise damage output, as that is a safe attack window. During the slam attack, the agent attacks during the jump, so the agent does not have to wait for the attack to be over. During the Leaping Bludgeon, the agent passes underneath and attacks upwards with a jump. This allows the agent to deal more damage while waiting for False Knight to land again. During the Rage, the agent attempts to hit in-range falling rocks towards False Knight. When staggered, the False Knight's maggot is the sole target of the agent. The agent will continuously attack when possible, and move inwards if necessary to maintain proper spacing.

Limitations The ReactiveKnowledgeAgent has some strategic mistakes, causing the agent to take more damage on certain False Knight attacks. The agent usually takes damage on Rage and Leaping Bludgeon. During Rage, the agent rushes in too far, being too focussed on finding more falling rocks to hit towards False Knight. During Leaping Bludgeon, the agent is too greedy for more upwards hits, tending to take damage near the end of the leap. Furthermore, during the Leap attack, the agent does not move out of the way enough if the Leap happens to land on top of the player. Finally, the agent does not take falling rocks into account in other attacks than Rage, which occur during phase 2 and 3 of the fight. This also leads to more damage.

4.4.10 AdaptiveKnowledgeAgent

Design Philosophy The AdaptiveKnowledgeAgent should have an improved strategy compared to the ReactiveKnowledgeAgent. The agent should have its movement changed to adapt to the attacks and different situations within those attacks. Using better positioning, more advanced attack strategies to more efficiently use attack windows, this agent would be able to perform a lot better against False Knight.

Target boss The AdaptiveKnowledgeAgent was built for False Knight.

Core strategy The AdaptiveKnowledgeAgent is similar to the ReactiveKnowledgeAgent, with the exception of a new strategy for the Rage and Leaping Bludgeon attacks, and improved falling rock awareness during the Slam attack. During the Rage attack, the agent retreats to a corner of the arena, and waits for the attack to end, accepting only the risk of being hit by a stray falling rock, rather than risking more damage. For the Leaping Bludgeon attack, the agent calculates a safe point behind the False Knight's predicted landing position. If this point is valid, not too far into the corner of the arena or outside of it, the agent walks to it during False Knight's jump, then turns around to continuously attack False Knight. Finally, with the Leap attack, the agent will try to determine where False Knight is going to land, to determine if it should loosely follow as before, or move out of the way if the leap is towards the agent.

Positioning rules Similarly to the ReactiveKnowledgeAgent, the agent will be mainly positioning itself near False Knight, with the exceptions to a few attacks. Similar to the ReactiveKnowledgeAgent, this agent will try to walk underneath False Knight during the Leaping Bludgeon attack, but checks first if that is a safe option. If it is, the agent will turn around when arriving at the safe point, and slashes to reliably deal damage to False Knight. Should the safe point not be a valid option, the agent retreats from its position, standing far enough away to not get hit by the attack. On this spot, the agent waits for the attack to be over. For the Leap attack, the agent determines the direction and height of the jump. If the direction is away, the agent will follow loosely. If the leap is towards the agent, the agent will wait and try to determine with the jump height of False Knight if it will be landing on top of the player or jumping over. When jumping over, the agent will wait, and eventually follow loosely in that direction. If the agent is at the landing position, the agent will attempt to move out of the way quickly. With the Rage attack, the agent moves to the nearest corner, and waits out the attack. This only causes occasional damage from falling rocks. Finally, during Slam attack in phases two and three, the agent is made aware of falling rocks, prioritising the dodging of these rocks if they happen to fall above the agent.

Attack rules The agent's attacks are very similar to the ReactiveKnowledgeAgent. This agent does however focus less on hitting rocks towards False Knight during the Rage attack, opting for a safer strategy. The agent also attacks more frequently during the Leaping Bludgeon attack, allowing for more damage dealt.

Limitations The AdaptiveKnowledgeAgent's strategy for the Rage attack can be improved by focussing more on the falling rocks. Hitting these rocks at False Knight would be a safe way to deal more damage per slash during an otherwise unsafe attack. The agent would also improve if a dodging strategy was implemented to deal with falling rocks during the idle state of False Knight, as the falling rocks from Leaping Bludgeon only start to fall after the attack has finished. These changes would make the AdaptiveKnowledgeAgent better at dealing damage during the Rage attack, and safe during the later phases of the fight.

4.5 Experimental protocol

Each agent is tested by repeatedly fighting each boss. On each boss, each agent is made to fight the boss $100,000$ times, each time with a random seed. The boss's attacks will therefore differ for each trial. The RandomAgent also uses this random seed, so it always chooses different moves each trial. Each trial is sequential, but unrelated to the last, restarting positions and health back to the start for each new trial. Each agent, except for the RandomAgent, has a decision interval of 0.1 seconds, equivalent to one decision every six frames at the simulation's 60 frames per second. This means that an agent sticks to its decision for six frames before being allowed to make a new decision, rather than reacting to every individual frame, which produces more deliberate, stable behaviour.

5 Experiments

5.1 Experiment design

The experiments for this research are designed to test the agents' performance against the bosses outlined in Section 4.3. To evaluate the performances of the agents, several metrics are returned for each trial. The agents in these experiments are mostly designed for the target bosses mentioned for each of them in Section 4.4. Only the ReactiveKnowledgeAgent and the AdaptiveKnowledgeAgent are not tested on both bosses, however, as they use False Knight-exclusive knowledge that does not translate to Gruz Mother. These agents were given heuristics for being able to fight Gruz Mother, but these heuristics were taken directly from the HybridAgent as a fallback strategy if False Knight patterns are not recognised. It is therefore not informative to test these agents on Gruz Mother. In this section, the evaluation metrics are explained, the experiment structure is described, and the agents used for the experiments are stated.

5.2 Evaluation Metrics

To properly judge the effectiveness of each agent, several metrics are returned from each experiment. An agent performs well if the agent manages to survive for a longer time on a loss, and is

therefore not dying very quickly, or if the agent survives for a short time on a win, and is therefore quickly able to defeat the boss. Additionally, if the agent has a higher damage dealt metric, it is more likely to have performed well. For each individual attempt, the following metrics are returned:

Survival Time: Measures the duration of the simulation in seconds. Since the simulation ends if the player or the boss is defeated, survival time should be interpreted differently depending on the outcome. On a loss, a longer survival time indicates the agent lasted longer before being defeated, which is a positive indicator. On a win, shorter survival time indicates the agent defeated the boss more efficiently. For this reason, survival time on wins and losses should be considered separately. Survival time is in a range from 0 to 180, a reasonable maximum time, after which the simulation ends in a loss for the agent.

Damage Dealt: Measures the amount of damage the agent dealt to the boss during a simulation. This allows the agents to be compared in damage output, where a higher value means the agent managed to get closer to victory. This metric ranges from 0 to the boss's health. If no damage is dealt during False Knight's stagger, False Knight stands back up and regains its lost health for that phase, allowing the agents to gain infinite damage dealt. A higher value does therefore not guarantee a win in that fight. In the Gruz Mother fight, the maximum value is simply tied to Gruz Mother's maximum health.

Damage Taken: Measures the amount of damage taken by the agent. This shows how successfully an agent deals with incoming attacks, measuring defensive performance. A higher value means that the agent was taking more damage during the fight, and is therefore a less careful fighter. This metric falls in a range of 0 to 5, where having taken 0 damage is perfect performance, and taking 5 damage means the player was defeated, and the agent lost.

Damage Avoidance: A derived metric used in overview plots in Figure 9 and Figure 15, where it is labelled as "Low Dmg Taken". This metric is calculated as the inverse mean of damage taken as seen in Equation 1. A higher value indicates better defensive performance, and a lower value indicates more incoming damage. the +1 in the denominator prevents division by zero when no damage is taken.

$$\frac{1}{\text{Mean damage taken} + 1} \tag{1}$$

Player Health Remaining: Measures how many masks the agent had remaining. This metric is a reflection of the Damage Taken metric, where more damage taken is in direct correlation with health remaining.

Boss Health Remaining: Measures how much health was remaining after the simulation ended. Similarly to Damage Dealt, this is in a range from 0 to Gruz Mother's maximum health for the Gruz Mother fight. In the False Knight fight, the range is from 285, False Knight's total armour health and maggot health added together, to −5. While False Knight's health being depleted to 0 is enough for a win, if False Knight has 5 health remaining and then is hit by a launched falling rock, 10 damage is dealt to False Knight, leaving him with −5 health.

Win: Takes note of the outcome of the simulation in a binary score. A one means the simulation was a win for the agent, and a zero means a loss for the agent.

Using these metrics over thousands of iterations should show interesting patterns arise like the agents' win rates, the mean survival times, the mean damage output, the mean damage taken, and the mean time to defeat the boss. These calculated metrics give more insight into the performances of each agent.

5.3 Experimental setup

The experimental setup is based on $100,000$ trials per agent per boss. The number of trials was chosen to provide statistical stability, ensuring reliable means and win rates for each agent. Each trial uses a random seed, ensuring each trial will differ from the previous, ensuring varied boss behaviour across trials. After a trial is completed, the next one starts, with the positions and health values reset to their original values. This allows each trial to be a separate boss encounter, and there is no effect on other trials. Every agent, except for the RandomAgent, has a decision interval of 0.1 seconds, which means that an agent will repeat a decision made every frame until those 0.1 seconds have passed, and a new decision can be made. This decision interval was implemented to ensure agents make decisions that are more permanent, and agents cannot change their decision every frame, which would result in erratic behaviour. The agents tested per boss are shown in Table 4.

Gruz Mother	False Knight
RandomAgent	RandomAgent
SimpleAgent	SimpleAgent
AvoidantAgent	AvoidantAgent
HybridAgent	HybridAgent
PredictiveAgent	PredictiveAgent
	ReactiveKnowledgeAgent
	AdaptiveKnowledgeAgent

Table 4: The selection of agents tested per boss

6 Results

The results from the $1,200,000$ total trials have been plotted in several different views to outline different evaluation metrics.

6.1 Gruz Mother

For Gruz Mother, there were $500,000$ trials total. The first metric that is interesting is the win rate. This shows how many of the $100,000$ trials an agent was able to win. The win rates for the agents fighting Gruz Mother can be seen in Figure 4. The RandomAgent did not succeed to win a single trial, showing that some skill is required to beat Gruz Mother.

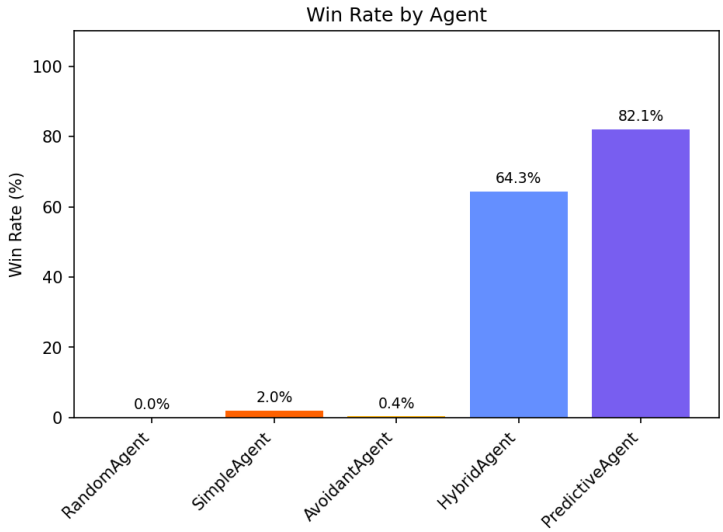


Figure 4: Win Rates for agents fighting Gruz Mother.

Most notably are the agents that do manage to win. The HybridAgent achieved a win rate of 64.3%, which is much higher than its predecessors. The PredictiveAgent is also interesting, as the agent manages to beat the HybridAgent by almost 20%, which shows another large gap in skill.

Another evaluation metric is survival time. Figure 5 shows a histogram of survival time for each agent.

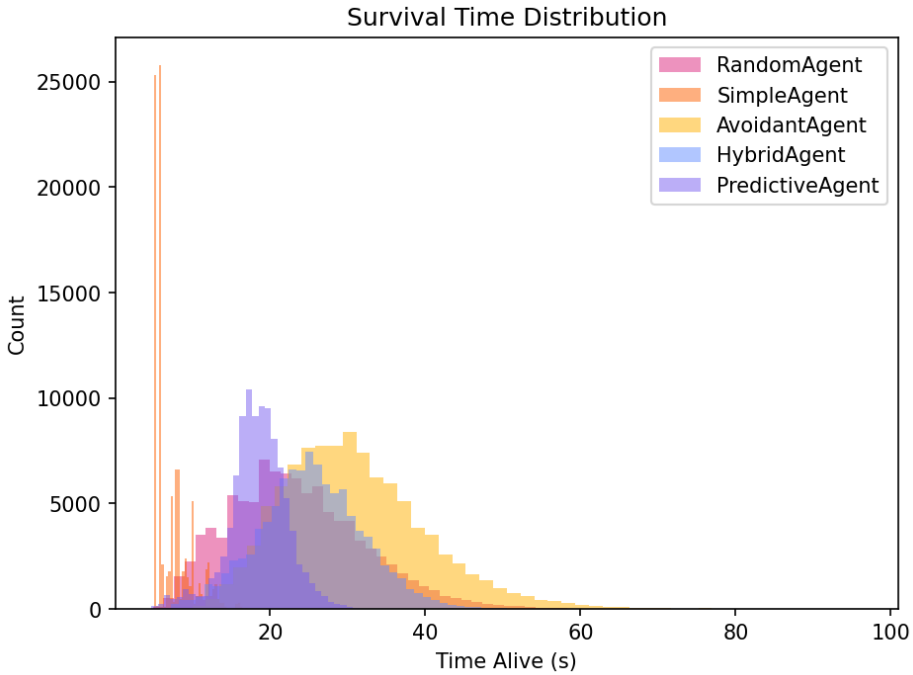
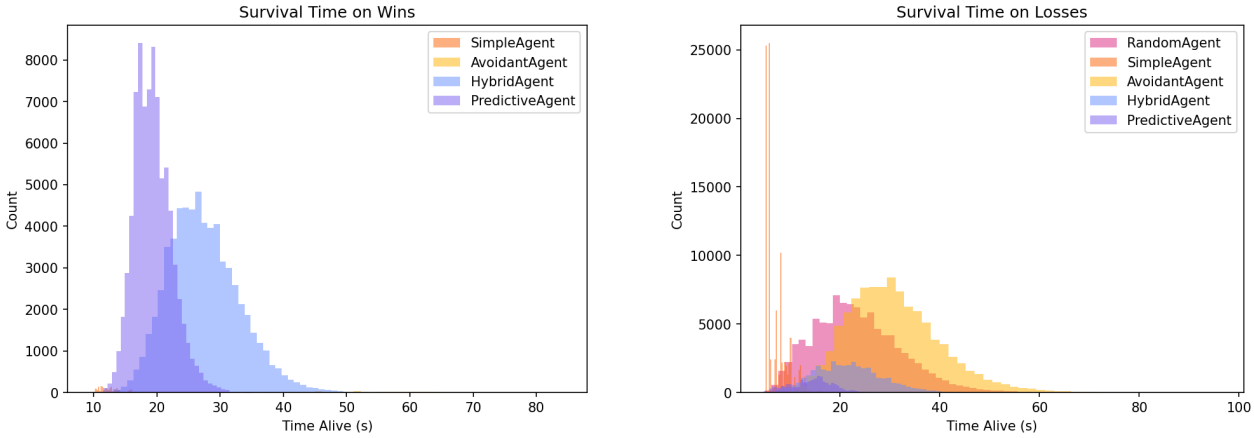


Figure 5: A histogram showing the survival times of each agent.

As mentioned in Section 5.2, survival time needs to be interpreted in differing ways depending on if the trial was a win or a loss. As a result, the histogram has been split into a survival time histogram for wins, and a survival time histogram for losses. This way, both interpretations of survival time can be considered. Both histograms can be seen in Figure 6.

As seen in Figure 6a, there are four agents to consider. Most notable are the SimpleAgent and the PredictiveAgent. The SimpleAgent does not win often, but has the lowest minimal time to defeat Gruz Mother, and the lowest mean. The PredictiveAgent is the second-best agent, looking at the fastest time to defeat Gruz Mother. PredictiveAgent is a lot more consistent, showing a good balance between speed and safety.

Figure 6b shows the survival times on losses. Here, a longer survival time is better, showing a better dodging strategy. The AvoidantAgent performs the best here, managing to survive the longest. The RandomAgent performs second best, closely followed by the HybridAgent. The SimpleAgent performs the worst by far.



(a) The histogram of wins.

(b) The histogram of losses.

Figure 6: Histograms of wins and losses, showing both interpretations of survival time.

Figure 7 shows the damage dealt and taken by each agent, normalised to a percentage. In this plot, a lower damage taken is showing a safer performance, and a higher damage dealt shows a more effective performance. The most notable agents are the HybridAgent and the PredictiveAgent. Both agents manage to deal more damage percentage-wise to the boss than they take. SimpleAgent is also noticeably dealing more damage than RandomAgent and AvoidantAgent.

Another interesting evaluation plot is Figure 8, where the performance on wins and losses is shown. The left plot shows the mean boss health remaining on a losses; the lower this value, the closer the agent came to winning. This shows the HybridAgent and PredictiveAgent beating the other agents by a large margin. The PredictiveAgent has a slightly lower mean than the HybridAgent, making PredictiveAgent the better performing agent here.

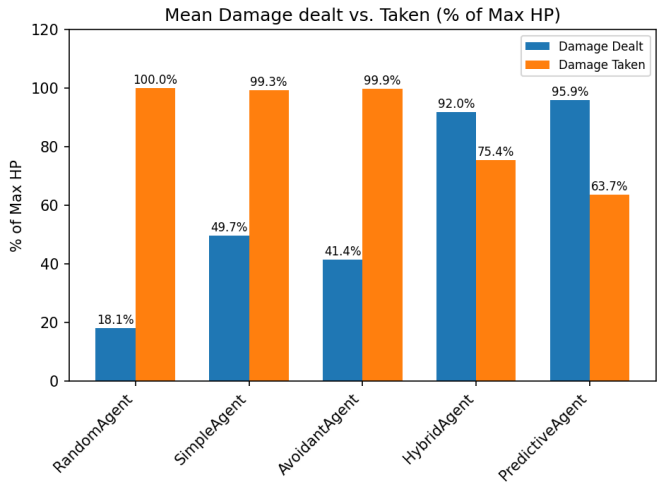


Figure 7: The mean damage dealt vs. damage taken, normalised to % of maximum health.

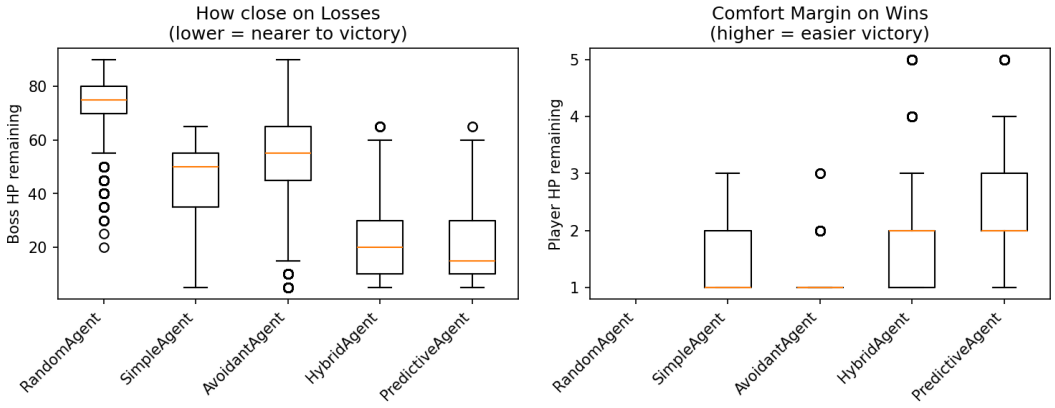


Figure 8: (Left) How close an agent got to defeating Gruz Mother on a loss. (Right) How comfortably an agent defeated Gruz Mother.

In the same figure, but on the right, the plot shows the mean remaining player health after defeating Gruz Mother. Here, PredictiveAgent performs the best, followed closely by HybridAgent. RandomAgent performed the worst, as it got no wins at all.

Finally, an overview of the data is shown in Figure 9. This plot shows the different important metrics in one plot, including the win rate, hp on win, survival time on wins, damage dealt, and damage avoidance. The metrics are normalised to show the best performing agent at 1.0, and other agents at the fraction they performed at of that best performing agent. For most metrics, the PredictiveAgent is the best performing agent, with HybridAgent being the overall second-best agent. The survival time metric is slightly different, as mentioned before, a lower score is better, as on a win, the agent should spend less time beating the boss, showing a more efficient strategy. The damage avoidance, labelled as "Low Dmg Taken" on the plot, is defined in Section 5.2.

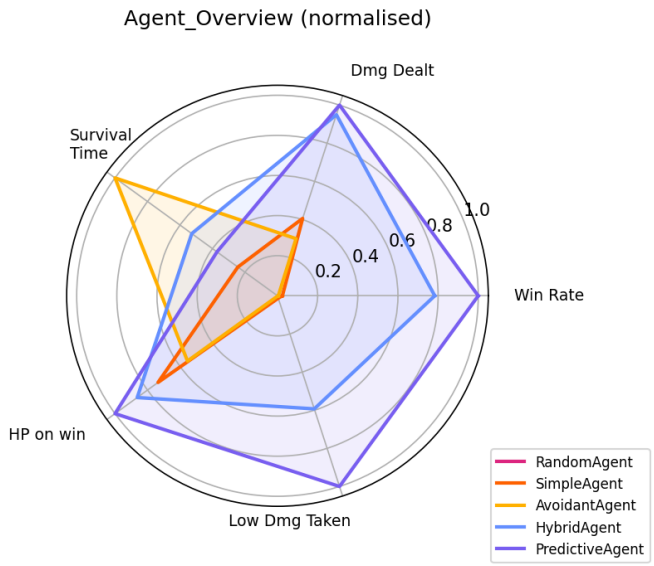


Figure 9: A radar plot, summarising important evaluation metrics.

6.2 False Knight

For the False Knight fight, there were 700,000 trials total, and 7 agents tested. The win rates here show that False Knight is a lot more difficult than the Gruz Mother fight. The win rates for each of the agents is shown in Figure 10.

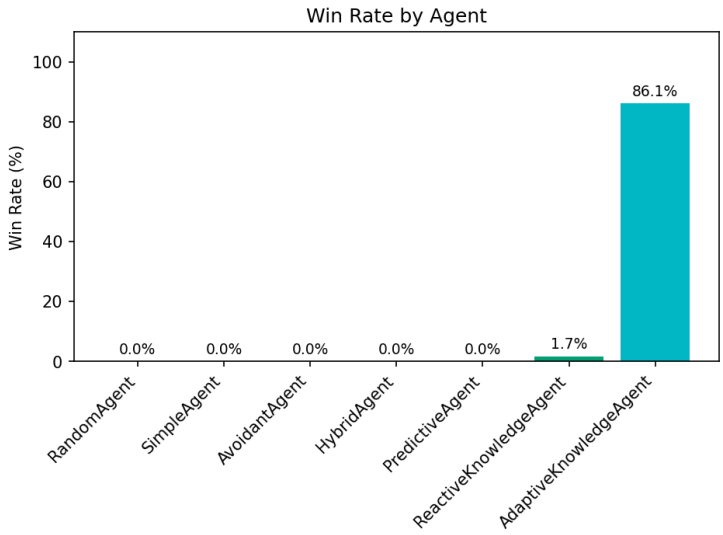


Figure 10: Win Rates for agents fighting False Knight.

Figure 10 shows that all of the agents built for Gruz Mother did not manage to win once. The ReactiveKnowledgeAgent is the first agent to win a few trials, but the AdaptiveKnowledgeAgent is able to win a lot more, achieving a win rate of 86.1%.

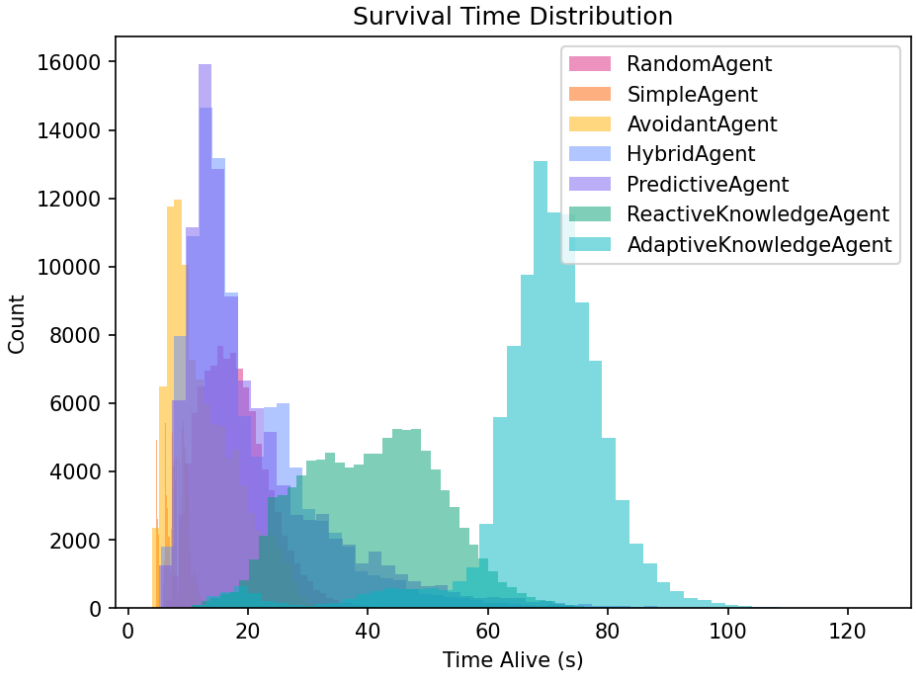
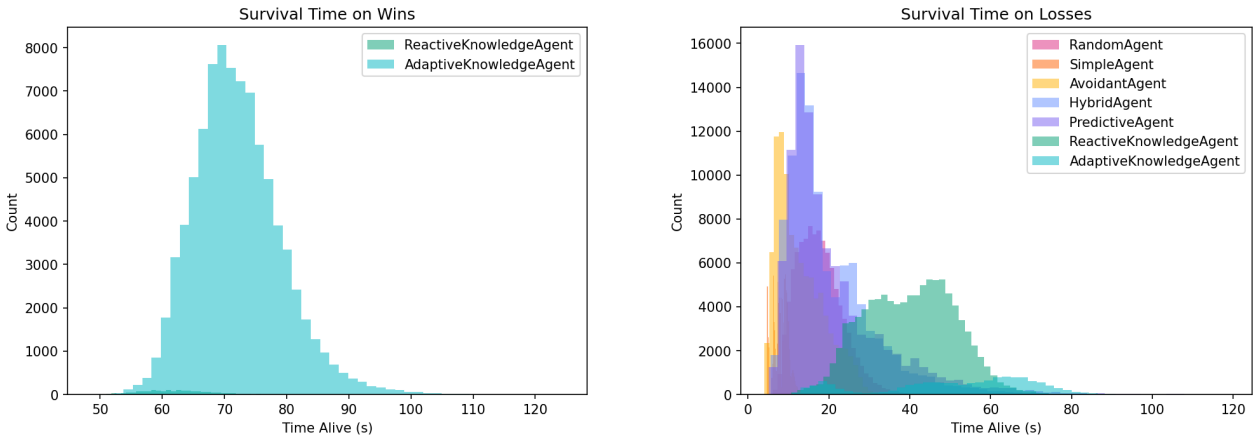


Figure 11: A histogram showing the survival times of each agent.

The survival times for the agents are shown in Figure 11. Most of the agents have a low mean survival time, showing that the fight is finished quickly for most agents.



(a) The histogram of wins.

(b) The histogram of losses.

Figure 12: Histograms of wins and losses, showing both interpretations of survival time.

As seen in Figure 12, most agents are finished with the fight against False Knight quickly because most agents don't survive for very long. In Figure 12a, both agents are roughly as efficient at defeating False Knight, with the ReactiveKnowledgeAgent being slightly faster, but winning a lot less often. In Figure 12b, the best agents are again the AdaptiveKnowledgeAgent and the ReactiveKnowledgeAgent. The next most interesting agent is the HybridAgent, able to survive the

longest on average among the other agents.

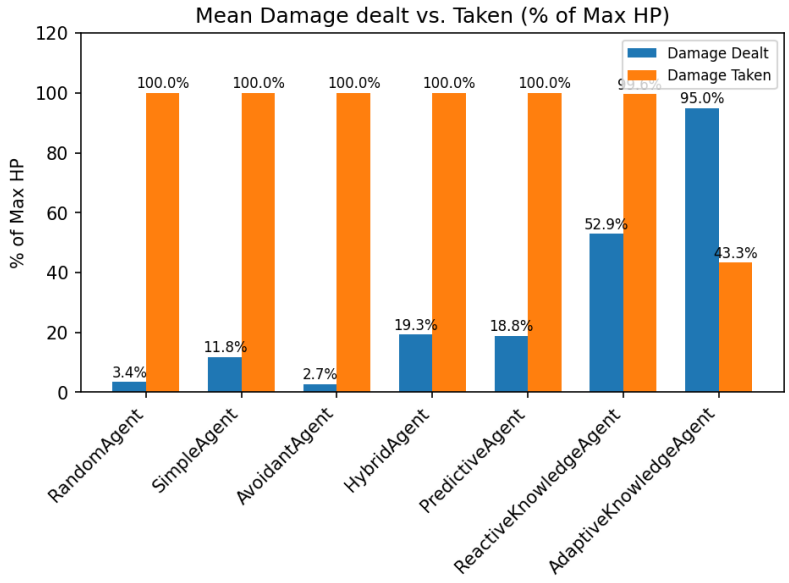


Figure 13: The mean damage dealt vs. damage taken, normalised to % of maximum health.

In Figure 13, the normalised damage taken and damage dealt is shown. No agents manage to deal more damage to False Knight on average than they receive, percentage-wise. The AdaptiveKnowledgeAgent gets close, and performs the best here, with ReactiveKnowledgeAgent in second place. Of the remaining agents, the HybridAgent again deals the most damage to False Knight on average.

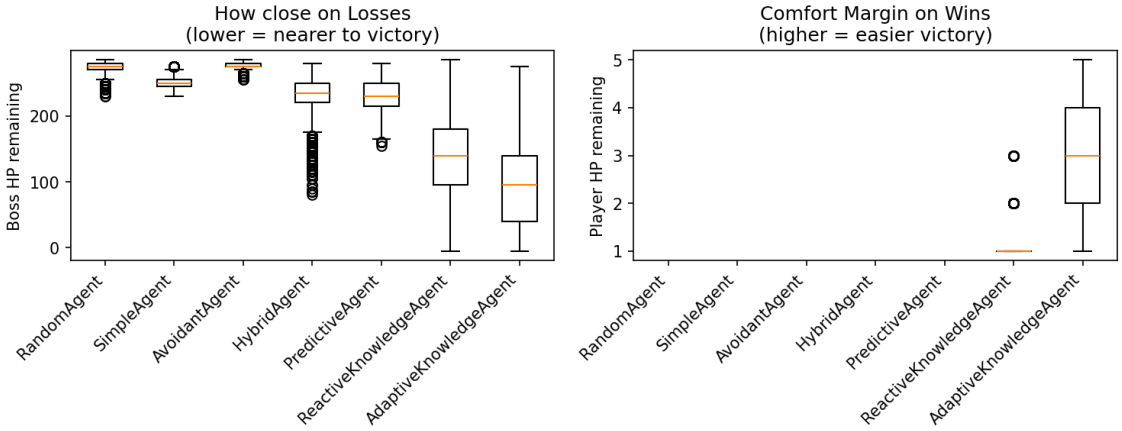


Figure 14: (Left) How close an agent got to defeating False Knight on a loss. (Right) How comfortably an agent defeated False Knight.

Figure 14 shows on the left how close the agent got to defeating False Knight on a loss. Again, AdaptiveKnowledgeAgent and ReactiveKnowledgeAgent perform the best. The HybridAgent has a lot of outliers to lower values of boss health remaining than the other agents. On the right,

the plot shows how most agents cannot beat False Knight, and therefore have no data. The ReactiveKnowledgeAgent can barely defeat False Knight, usually having 1 mask of health left after the fight. The AdaptiveKnowledgeAgent also usually has 1 mask of health left, with some more occurrences of higher health.

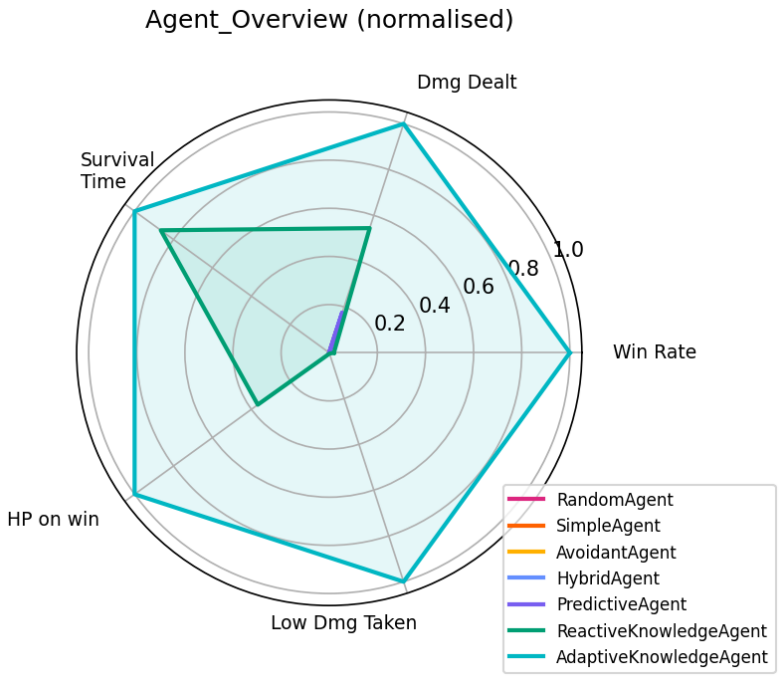


Figure 15: A radar plot, summarising important evaluation metrics.

Finally, the radar plot in Figure 15 shows a summary of the agents. Overall, the AdaptiveKnowledgeAgent performs the best. The ReactiveKnowledgeAgent is second best and the best in Survival Time, as it is slightly more efficient at defeating False Knight, when it does. Interestingly, the PredictiveAgent and HybridAgent are visible on this figure, only in the damage dealt metric. HybridAgent only slightly deals more damage than the PredictiveAgent, but compared to the AdaptiveKnowledgeAgent, the difference is not much.

7 Discussion

7.1 Gruz Mother

In the Gruz Mother fight, the RandomAgent unsurprisingly does not perform well, as the boss fights are supposed to be a test of skill. The SimpleAgent performs a lot better, being able to win, but not very often. The strategy of getting as close as possible to deal as much damage as possible only works if Gruz Mother stays slightly above the SimpleAgent, and the agent can simply attack upwards until Gruz Mother is defeated. This is uncommon, and therefore does not result in many wins for the SimpleAgent. On losses, the survival time is very short, as the agent decides to walk into the hitbox of Gruz Mother, taking damage nearly instantly after the invincibility timer has

passed. This gives the agent very large distinct peaks near the left of Figures 5 and 6b, which is about as fast as an agent can die.

The AvoidantAgent also does better than the RandomAgent, but worse than the SimpleAgent in wins. The AvoidantAgent does not deliberately attack Gruz Mother, only to get out of the corner when backed up enough. This leads to a longer survival time, but barely any success in defeating the boss. The AvoidantAgent has the longest survival time, both in wins and losses. The AvoidantAgent's strategy of dodging attacks is not flawless, and often the agent takes damage during a corner escape, eventually leading to a loss most of the time.

The HybridAgent is a large jump in performance compared to the previous agents. The agent achieved a win rate of 64.3%, substantially higher than the SimpleAgent with 2.0%. The HybridAgent shows that positioning is very important for fighting Gruz Mother. It is important to find a good balance between keeping enough distance to not get hit constantly like the SimpleAgent, but so much distance that no damage can be done, like the AvoidantAgent. The HybridAgent also implemented jump attacks, allowing the agent to deal more damage in a shorter amount of time. The agent does rely on the jump attack a lot, interfering with horizontal movement.

The PredictiveAgent performed the best overall on the Gruz Mother fight. The highest win rate of 82.1% is another large jump compared to the HybridAgent. The PredictiveAgent does have a lower mean survival time, in wins and in losses. This means the PredictiveAgent is more efficient on wins, but does lose quicker too. The agent deals more damage on average than any other agent, and takes the least damage on average. The PredictiveAgent uses the predictions of Gruz Mother's movement to effectively decide how to dodge out of the corner, allowing the agent to take less damage during the fight. Additionally, the agent is less inclined to using the jump attack compared to the HybridAgent, allowing it to keep its distance more effectively, taking less contact damage. The PredictiveAgent is therefore overall the best agent, even if it doesn't survive the longest on losses.

7.2 False Knight

The agents made for Gruz Mother perform terribly against False Knight. These agents obtain a win rate of 0.0%. They do not have the understanding of False Knight's attacks. They also do not deal enough damage to stagger False Knight without taking too much damage. Interestingly, the best agent for Gruz Mother, the PredictiveAgent, did not perform the best against False Knight. The HybridAgent performed slightly better, surviving for longer than the PredictiveAgent, and dealing slightly more damage on average. This is due to the reliance on the jump attack, which allows the HybridAgent to dodge the Slam attack for example, avoiding two masks of damage. This dodging is not consistent, as it is not based on when the Slam is done. The PredictiveAgent was built to be less reliant on the jump attack, allowing for better horizontal positioning, but in the False Knight fight, that comes at the cost of taking more damage from Slam attacks.

The first agent able to win occasionally is the ReactiveKnowledgeAgent. Giving the agent knowledge of False Knight's attacks and states allows the agent to dodge more effectively during attacks. The agent can more reliably jump over the Slam attack, taking less damage. During the Rage attack,

this agent tends to take most of its damage, together with the Leaping Bludgeon attack. During Rage, the agent does not give enough priority to staying outside of the False Knight's attacks in the middle, often taking damage when looking for falling rocks to hit towards False Knight. During the Leaping Bludgeon attack, the agent can get a bit too greedy, often getting too close to False Knight and overlapping with his hitbox, taking contact damage. The ReactiveKnowledgeAgent is also not taking falling rocks into account in attacks other than Rage, occasionally taking damage during those attacks in later phases. The ReactiveKnowledgeAgent is able to occasionally win, unlike the previous agents.

The AdaptiveKnowledgeAgent makes better use of the added game state information. The agent is more careful with falling rocks, giving priority to moving out of the way over the usual attack strategy. The AdaptiveKnowledgeAgent also has a new strategy against the Leaping Bludgeon attack, finding a safe area to stand, and walking to this spot. From this safe spot, the AdaptiveKnowledgeAgent is usually able to deal more damage to False Knight, making a stagger more likely to occur sooner. Being able to determine the trajectory of the Leap attack allows the agent to take less damage, not waiting for False Knight to land on top of it. Finally, the Rage behaviour is to stand in a corner and wait for the attack to be over, only ever getting hit by the occasional falling rocks. These behaviours allow the AdaptiveKnowledgeAgent to survive for longer than any other agent, on wins and losses. The AdaptiveKnowledgeAgent also deals the most damage on average, progresses further into the fight than any other agent on losses, and achieves the highest win rate of 86.1%. This makes this agent the best overall for the False Knight fight.

7.3 Further Research

This research demonstrates that heuristic agents can be effective for scripted boss encounters, but several directions remain for further investigation.

Simulation-based approaches such as Monte Carlo Tree Search (MCTS) offer an alternative approach based on stochastic forward simulation. MCTS has demonstrated strong performance in complex adversarial domains, most notably in the success of AlphaGo [13]. However, research has shown that MCTS performance is highly dependent on domain characteristics such as branching factor, reward sparsity, and stochasticity [16]. This introduces additional challenges, since the decision space facing each action is large and must be searched within whatever computational budget is allotted to remain practical for repeated, large-scale trials. Investigating whether MCTS operates effectively within this computational budget, given the branching factor of boss encounters, and how it compares to heuristic agents developed here, would be a natural next step.

Reinforcement learning offers another promising direction. Unlike heuristic agents which require domain expertise to design and do not adapt, an RL agent could discover effective strategies through experience instead of hand-crafted rules. This research manually designed agents with distinct behavioural strategies, namely aggressive, passive, balanced and knowledge-based. Barthet et al. [1] demonstrate that RL can generate agents with distinctive play styles automatically, by training on human demonstrations. Applying a similar approach to boss encounters could produce a wider variety of strategies than manual design allows, and potentially discover approaches a designer

would not think to encode.

Another natural next step is implementing a greater number of bosses to compare the agents with. Testing agents gainst a wider range of Hollow Knight-inspired bosses would allow evaluation across more diverse attack patterns and a good general strategy.

8 Conclusions

This research designed and evaluated six heuristic agents of increasing sophistication, tested against two Hollow Knight-inspired boss encounters implemented in a custom Pygame simulation. On the simpler encounter, Gruz Mother, the best-performing agent reached an 82.1% win rate, while on the more complex False Knight encounter, agents without domain-specific knowledge failed to win a single trial, and the best knowledge-based agent achieved an 86.1% win rate. These results show that performance depends heavily on both balanced general strategy and, for more complex encounters, domain-specific knowledge.

Measuring this performance required more than a single metric. By collecting win rate, survival time split by outcome, damage dealt, and damage taken across experiment trials, a more nuanced picture of agent behaviour emerged than win rate alone would allow. This answers the first research subquestion on how performance can be measured and compared. This multi-dimensional view revealed that agents can perform well on some metrics while failing on others: the AvoidantAgent, for instance, survived longer than most agents on losses despite rarely winning, a pattern a single win-rate metric would have hidden entirely.

This same data also surfaced clear trade-offs between competing objectives, answering the second subquestion. The SimpleAgent and AvoidantAgent represent two extremes of aggression and safety, and neither performs well as a result. Agents that balanced dealing damage against avoiding it, such as the HybridAgent, performed substantially better. A related trade-off emerged around the HybridAgent's jump attack: relying on it heavily improved damage output, but interfered with horizontal spacing. This was a limitation that the PredictiveAgent addressed by using it more sparingly, at the cost of weaker performance against False Knight, where the jump attack incidentally helped dodge the Slam attack.

Taken together, these findings answer the main research question: decision making strategies for scripted boss encounters can be optimised by iteratively identifying the limitations of a previous agent, and addressing them in the next. This progression moved from purely aggressive or purely defensive baselines, through balanced combinations of the two, to agents using full domain-knowledge to handle the specific demands of a more complex encounter. Notably, the HybridAgent's balanced approach proved the most generalisable strategy, performing the most competitively on both boss encounters without domain-specific knowledge, suggesting that balanced positioning and attack strategies form a strong foundation regardless of the specific encounter, one that can then be extended with targeted domain-specific knowledge where a boss's complexity demands it.

References

[1] Matthew Barthet, Ahmed Khalifa, Antonios Liapis, and Georgios N. Yannakakis. Generative personas that behave and experience like humans. September 2022. doi:10.1145/3555858.3555879.

[2] Ivan Bravi, Diego Perez-Liebana, Simon M. Lucas, and Jialin Liu. Shallow decision-making analysis in general video game playing. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, 2018. doi:10.1109/CIG.2018.8490365.

[3] David Churchill and Micheal Buro. Incorporating search algorithms into rts game agents. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 2–7, 2012. doi:10.1609/aiide.v8i3.12548.

[4] Micheal Dann, Fabio Zambetta, and John Thangarajah. Real-time navigation in classical platform games via skill reuse. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 1582–1588, 2017.

[5] Hollow Knight Wiki contributors. False knight. https://hollowknight.wiki/w/False_Knight, 2026. Accessed: June 2026.

[6] Hollow Knight Wiki contributors. Gruz mother. https://hollowknight.wiki/w/Gruz_Mother, 2026. Accessed: June 2026.

[7] Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D. Gaina, Julian Togelius, and Simon M. Lucas. General video game ai: A multitrack framework for evaluating agents, games, and content generation algorithms. *IEEE Transactions on Games*, 11(3):195–214, 2019. doi:10.1109/TG.2019.2901021.

[8] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Lariviere, Alina Beygelzimer, Florence d'Alche Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research(a report from the neurips 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164):1–20, 2021. URL: https://arxiv.org/pdf/2003.12206.

[9] Pichit Promsutipong and Vishnu Kotrajaras. Enemy evaluation ai for 2d action-platform game. In *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–6, 2017. doi:10.1109/JCSSE.2017.8025906.

[10] Erika Puiutta and Eric M. S. P. Veith. Explainable reinforcement learning: A survey. In Andreas Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar Weippl, editors, *Machine Learning and Knowledge Extraction*, pages 77–95, Cham, 2020. Springer International Publishing.

[11] Pygame Development Team. *Pygame: A Free Open Source Python Library (v2.6.1)*, 2024. URL: https://www.pygame.org.

[12] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3rd edition, 2016.

[13] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[14] K. Siu, E. Butler, and A. Zook. A programming model for boss encounters in 2d action games. *AAAI Technical Report WS-16-22*, pages 86–92, 2016.

[15] Skurry. Hollow knight but it's only the hitboxes. https://www.youtube.com/watch?v=DDKoy6XTfTA, 2023. YouTube video. Accessed: June 2026.

[16] Dennis J.N.J. Soemers, Guillaume Bams, Max Persoon, Marco Rietjens, et al. Towards a characterisation of monte-carlo tree search performance in different games. *arXiv preprint arXiv:2406.09242*, 2024. [cs.AI].

[17] Zhentao Tang, Yuanheng Zhu, Dongbin Zhao, and Simon M. Lucas. Enhanced rolling horizon evolution algorithm with opponent model learning: Results for the fighting game ai competition. *IEEE Transactions on Games*, 15(1):5–15, 2023. doi:10.1109/TG.2020.3022698.

[18] Team Cherry. Hollow knight, 2017. Video game. URL: https://store.steampowered.com/app/367520/Hollow_Knight/.

[19] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer Nature, 2 edition, 2024. URL: https://gameaibook.org/wp-content/uploads/2025/08/AI_and_Games_2nd_Edition.pdf.

[20] Weipu Zhang, Adam Jelley, Trevor McInroe, Amos Storkey, and Gang Wang. Object-centric world models from few-shot annotations for sample-efficient reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL: https://openreview.net/forum?id=qmEyJadwHA.